AMENDMENT TO THE SPECIFICATION

Please replace the paragraph on page 16, lines 10-19 with the following amended paragraph.

In one embodiment, the source code of the kernel code ~~section~~ sections is compiled automatically by one of several compilers corresponding to the available hardware accelerators. In an alternate embodiment, a programmer may manually realize the executable code from the source code of the kernel code sections, as shown by the dashed line from step 656 to step ~~550~~ 650. In a third ~~embodiment~~ embodiment, the source code of the kernel code sections is compiled automatically for execution on both the processors and the hardware accelerators, and both versions are loaded into the resulting binary code. In a fourth embodiment, a hardware accelerator is synthesized in a custom hardware accelerator description.

Please replace the paragraph on page 18, lines 1-17 with the following amended paragraph.

The TAPR of step 660 supports two subsequent steps in the preparation of the source code for execution. In step 664, a table is prepared for subsequent use by the run time system. In one embodiment, the table of step 664 contains all of the three-tuples corresponding to allowable combinations of designs (from step 656), bins, and variants. A variant of a design or a bin is any differing hardware implementation where the functional input and the outputs are identical when viewed from the outside. The variants of step 664 may be variants of memory separation, such as the separation of memory into upper and lower memory as discussed above. Other variants may include differing geometric layouts of MCPEs within a bin, causing differing amounts of clock delays being introduced in the microcodes, and also whether or not the use of various parts of the MCPEs overlap the use of the parts in other ~~varients~~ variants. In each case a variant performs a function whose inputs and outputs are identical outside of the function. The entries in the table of step 664 point to executable binaries, each of which may ~~each~~ be taken and executed without further processing at run time. The table of step 664 is a set of all alternative execution methods available to the run time system for a given kernel section.

Please replace the paragraph on page 21, lines 9-15 with the following amended paragraph.

In the Figure 7B embodiment, processor A 752 or processor B 770 may execute the overhead code identified in step 652 and created as object files in step 672 of the Figure 6 process. DSP 0 756, DSP 1 758, and DSP 2 760 may execute the kernel code identified in step 652. Each processing element processor A ~~702~~ 752 and processor B ~~720~~ 770 is supplied with an instruction port, instruction port ~~724~~ 774 and instruction port ~~722~~ 772, respectively, for fetching instructions for execution of overhead code.

Please replace the paragraph on page 26, lines 6-8 with the following amended paragraph.

In the exemplary Figure 9C emobodiment, RTK 704 loads the first overhead code 910, ~~960~~ 950 sections into processor 1 702 and processor B 720, respectively, for execution in time periods 980 and 962, respectively.

Please replace the paragraph on page 26, lines 6-8 with the following amended paragraph.

In Figure 10 embodiment, it should be noted that it is not necessary that there be an valid entry at each location of the matrix. There are situations where there are relatively few valid sets of designs, bins, and ~~varients~~ variants. These situations give rise to a sparsely-populated TAPR matrix. In other situations, there may be a valid set of designs, bins and ~~varients~~ variants for all locations in the matrix. These situations give rise to a fully-populated TAPR matrix.